# nova230 for Modbus

## User's Manual

7001027003 S8

This description corresponds to the current protocol, EPROM 501143.001 Index b, and to Version 1.6 of the parameterisation programme, ModbusPara230

# Table of Contents

## Trademarks

| | |
|---|---|
| Designer | Trademark of Micrografx, Inc. |
| Micrografx Designer | Trademark of Micrografx, Inc. |
| Media Manager | Trademark of Micrografx, Inc. |
| Windows | Trademark of Microsoft Corporation |
| Microsoft Office 97 Professional | Trademark of Microsoft Corporation |
| MS Office | Trademark of Microsoft Corporation |
| Microsoft Access 97 | Trademark of Microsoft Corporation |
| Microsoft Office 2000 | Trademark of Microsoft Corporation |
| Microsoft Word | Trademark of Microsoft Corporation |
| Acrobat Reader | Adobe Systems Incorporated |
| Pentium | Trademark of Intel Corporation |

# 1 General

## 1.1 Description of the system

The nova230 is a compact automation station in the EY3600 system family, with an interface function. The nova230 for Modbus (EYL 230 F040 - Compact AS Modbus RTU) allows the connection of Modbus components via the serial interface.
It is used, on the one hand, for controlling in HVAC engineering and on the other, it has an external connection interface (RS232 interface in the form of a DB9 connection) via which Modbus system data can be received or sent.

The AS section has a total of 28 hardware inputs and 10 hardware outputs. All in all, this automation station has 256 MFAs (Machine Fine Addresses), which are assigned as follows:
- 35 MFAs: hardware inputs / outputs (range MFA 0...59)
- 4 MFAs: service addresses (MFA 60...63)
- 1 MFA: communication fault (alarm) (MFA 255 – Bit 31)

191 (freely parameterisable) software MFAs are available for data transfer with the external system.

The nova240 Control Panel is connected to the station via an RJ45 socket, and it can be recessed in the cover on the AS nova230. The Control Panel allows handling and visualisation of all data (except for the HDB) from the station (read measured values, alarms and status, change setpoints and output positioning commands).

The nova230 can read out data from the Modbus components and send data to them. The desired data are interrogated cyclically (about 10 telegrams per second) by the Modbus network. Commands to the Modbus bus system are given preference and are transmitted without a delay.
At present, most of the simple datapoint types and function codes can be processed. A list of all supported datapoint types and function codes is given in section 3.
In order to parameterise the transfer list (datapoint list), a parameterisation programme with a download function is used. The 'ModbusPara230' parameterisation programme is an Excel template with Visual-Basic macros, used to
- create the datapoint list,
- check the parameters that have been entered, and to
- create an Intel Hex File followed by download into the nova230.

The data are transferred (downloaded) to the nova230 via the serial interface of the PC.

A possible communication fault between the nova230 and the Modbus devices is signalled to the management level via MFA 255.
MFA 255 - Bit 31 = 0 = communication error (alarm)

For commissioning purposes, it is possible to log the communication between the nova230 and the Modbus devices using terminal software on the PC.

## 1.2 System structure during operation

Most Modbus devices have an RS485 interface and they can be integrated into a network with a maximum of 32 RS485 Modbus devices (without repeaters). Each device has a unique device address (slave ID). Since the nova230 only has one RS232 interface, it is necessary to use an RS232/RS485 interface converter. Electrically isolated interface converters are advisable, and these are mentioned in Annexe 6.4.1.
If only one device is in use and the Modbus RTU allows an RS232 connection, it is possible to select a point-to-point connection.
The nova230 is always the Modbus Master and it only supports transmission mode RTU. The maximum transmission speed is 9600 Baud (since Version 1.6 theoretically 19'200 as well) and 8 data bits should be used. Broadcast telegrams (with address ID 0) are not supported.

The connection diagrams for the two Modbus topologies that are supported are shown below.

### 1.2.1 Point-to-point connection via RS232

The nova230 is specified as DTE (Data Terminal Equipment) and it must be connected to the external device as follows:

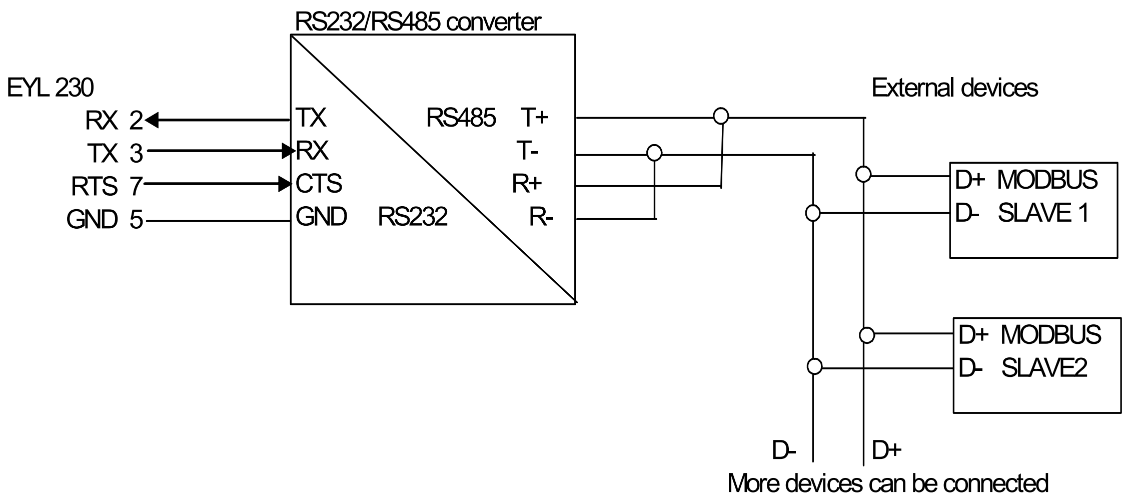
```
EYL230 socket RS232                External device

Pin 2   =     Rx  ---------------------------------- Tx
Pin 3   =     Tx  ---------------------------------- Rx
Pin 5   =    GND  ---------------------------------- GND
Pin 7   =    RTS  ----------(optional)------------   CTS
```

The connection between the RTS (Request to Send) line and the CTS (Clear to Send) line of the external device is optional; it may be necessary depending on the Modbus device.

## 1.2.2 Bus connection via RS485

An RS485 bus can operate in two different modes: Full Duplex (transmission with 4 data lines in two directions at the same time) or Half Duplex Mode (transmission with 2 data lines in both directions, but consecutively). For the latter option, a handshake signal (CTS and RTS) is needed.
In the case of the nova230, Half Duplex Mode has to be used. This means that the transmitter (T-Transmitter) and receiver (R-Receiver) must be connected to each other as follows:

T- with R- and T+ with R+.



This gives communication on 2 data lines.

More detailed information can be found in the manual for the RS232/RS485 converter and the Modbus specification [2].

# 2 ModbusPara230 installation

## 2.1 Software requirements

To install ModbusPara230, the following software must be installed on your PC already:
- Windows 98 with Office 97 SR2 **or**
- Windows 2000 with Office 97 SR2 / Office 2000
- Windows XP with Office XP

## 2.2 Programme description

ModbusPara230 (**ModbusPara230.xls**) is an Excel template with Visual-Basic macros used to
- generate the datapoint list,
- check the parameters that have been entered, and to
- create an Intel Hex File followed by download into the nova230.

The programme package contains these files:

DISK1:

| | |
|---|---|
| - REGSVR32.EXE | Registration programme for ActiveX components |
| - DTB.EXE | Text-Binary converter |
| - BINHEX.EXE | Binary-Hex converter |
| - DTB.PIF | Programme information file |
| - BINHEX.PIF | Programme information file |
| - INSTALL.XLS | Installation programme |
| - MSCOMM32.OCX | ActiveX control element for communication |
| - MSCOMCTL.OCX | ActiveX control element for the progress bar |

DISK2:

| | |
|---|---|
| - MSCOMCTLLIB.TWD | Library |
| - ModbusPara230.XLS | Parameterisation template |
| - ModbusPara230.Doc | Documentation file |

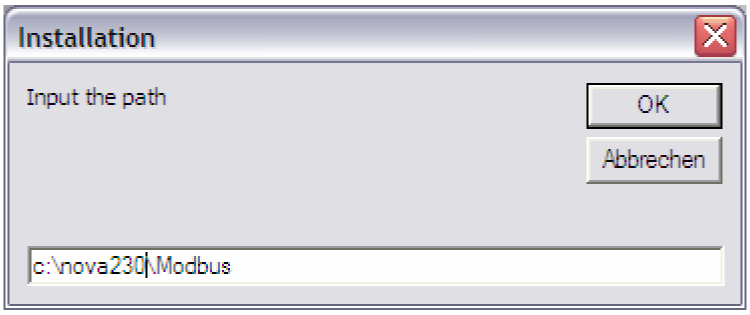
**Note:**
The following DOS programmes are needed for the automatic generation of the Intel Hex file:

| | | |
|---|---|---|
| 1. | DTB.EXE | DomToBin converter |
| 2. | BINHEX.EXE | Binary to Hexfile converter |

These programmes must be stored in the same directory as the Excel file, since they are started by the Excel application from this directory.
The name structure of the directory must not contain more than 8 characters (DOS conventions). If one of these files is missing or if the name structures of these directories are larger than 8 characters, the programme cannot be installed and the Hex file cannot be created. (→ Error message).

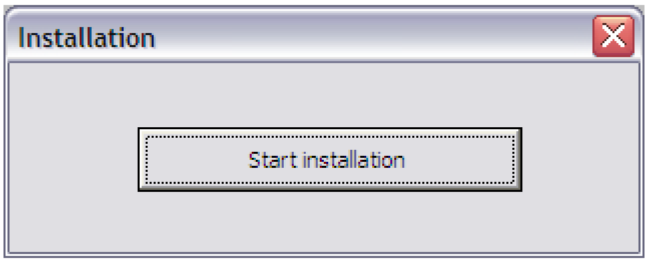
## 2.3 Installation

- Copy the files into a temporary directory (e.g. c:\temp\Modbus). The directory must not contain any blank spaces.
- Start the installation by opening file **INSTALL.XLS**
  (Excel – File/Open/INSTALL.XLS or
  Windows Explorer – double-click on INSTALL.XLS)
- Click on 'Enable macros'
- Enter the installation path manually.
  (Default setting: c:\novaCom\Modbus230).
  **The name of the directory must be changed, because the DOS structure used means that no more than eight characters may be used.**

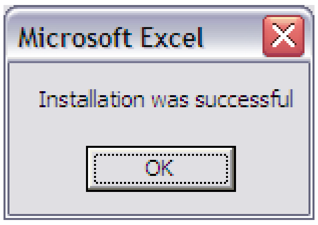


c:\nova230\Modbus

Once you have changed the programme path, you can start the installation by pressing the **'Start installation'** button.



The installation is complete.



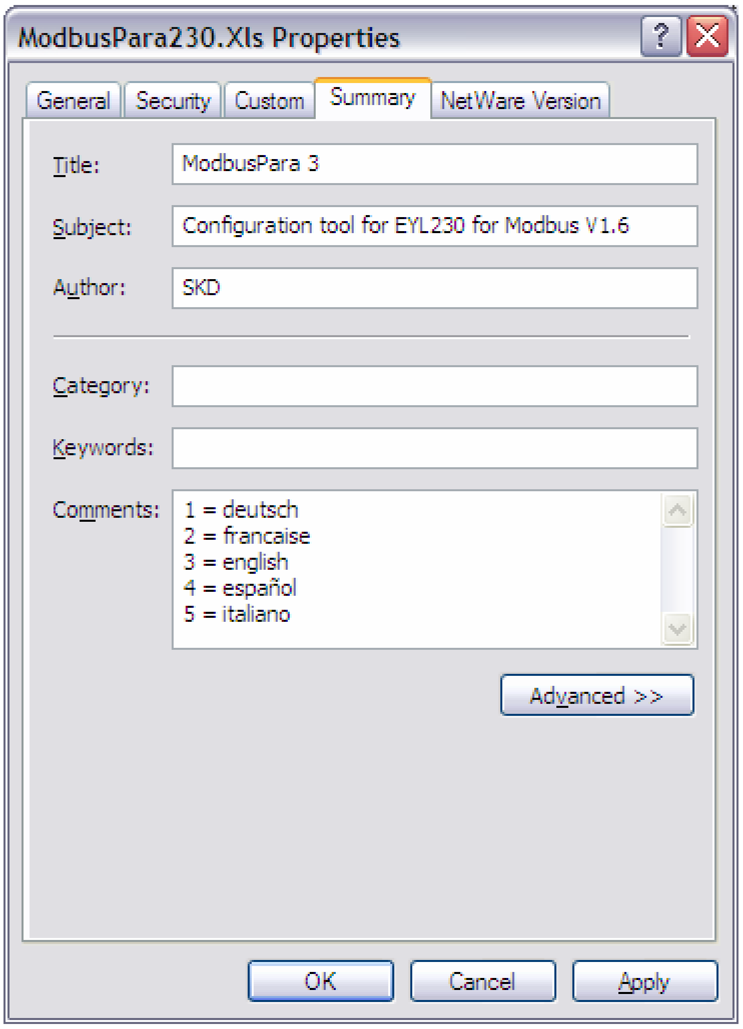
Installation was successful

**Note:**
The installation must be carried out via the installation programme. It is not sufficient simply to copy the files into a programme directory.

## 2.4 Language settings

The default language version set on the ModbusPara230 software is **English**.
The language attributes can be changed in the Properties menu in file **ModbusPara230.xls**.

- In Windows Explorer, click on file **ModbusPara230.xls** with the right-hand mouse button and select **Properties.**
- In the Properties dialogue of ModbusPara230.xls, go to the Fileinfo tab
- You can set the language variant you want at Description/title.



Title: ModbusPara 3
Subject: Configuration tool for EYL230 for Modbus V1.6
Author: SKD
Category:
Keywords:
Comments: 1 = deutsch
2 = francaise
3 = english
4 = español
5 = italiano

The following language attributes are available:

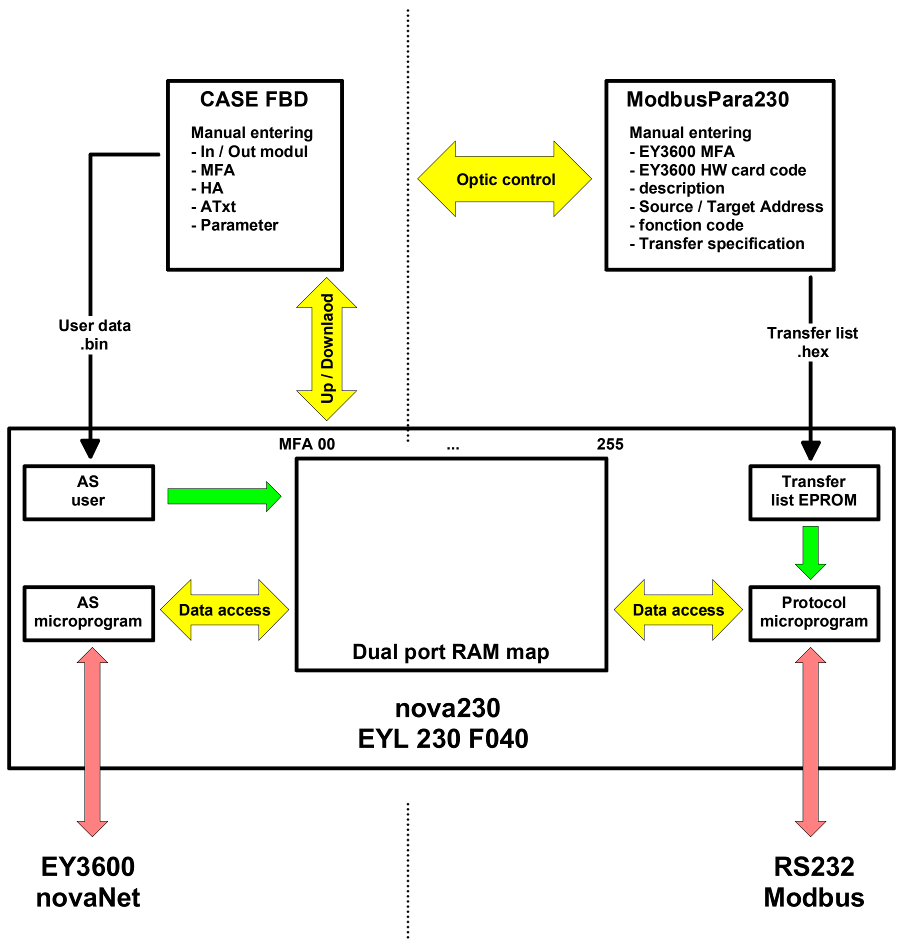
| | |
|---|---|
| - ModbusPara 0 | Texts can be entered (for internal use only) |
| - ModbusPara 1 | German |
| - ModbusPara 2 | French |
| - ModbusPara 3 | English |
| - ModbusPara 4 | Spanish |
| - ModbusPara 5 | Italian |

**Note:**
In the definition of the title, only one blank space must appear between the name and number of the ModbusPara.

# 3 Engineering

## 3.1 Philosophy

The Modbus datapoints are stored by the protocol micro-programme in an MFA (range 64...254) in the nova230-RAM map. The criteria are the engineered MFA and the corresponding card code. This MFA then undergoes further processing by the AS micro-programme like a normal EY3600 hardware datapoint.



So that this MFA behaves like an EY3600 datapoint, a parameterisation with the help of CASE FBD is required.

- set the module manually, and connect it if necessary
- assign the hardware MFA manually in accordance with the transfer list
- define the parameters: house address, etc.

This Modbus datapoint requires a parameterisation which can be performed with the help of the ModbusPara230 programme.

- manual assignment of the MFA in accordance with the CASE FBD Plan
- manual assignment of the hardware card code in accordance with the CASE FBD module and the Modbus function code
- Modbus target or source address

## 3.2 Engineering with CASE FBD

Parameterisation of the AS with CASE FBD is handled manually. Engineering via CASE project is not yet possible.
On every MFA that is parameterised as a transfer address, a hardware I/O module has to be set manually (in accordance with the hardware card code and the datapoint type as generated in the transfer list).

### 3.2.1 Relationship between input/output modules, datapoint type and function code

The following table shows the relationships between the respective hardware card codes, datapoint types and the corresponding CASE FBD input/output modules, and also the Modbus function codes:

| **Datapoint function** | **Hardware card code** | **CASE FBD modules** | **Datapoint type** | **Function code** |
|---|---|---|---|---|
| Measured value | $70_{hex}$ | AI | 2 to 9 | 3, 4 |
| Metered value | $D0_{hex}$ | CI | 2 to 9 | 3, 4 |
| Setpoint | $80_{hex}$ | AO | 2 to 9 | 3/6 |
| Setpoint with ARM | $A0_{hex}$ | AO | 2 to 9 | 3/6 |
| Alarm / status // BRM | $10_{hex}$ | BI (fC8) // DI (fCI) | 1 | 1, 2, 3[*)], 4[*)] |
| Switching command | $20_{hex}$ | DO (fCI) | 1 | 1/5 |
| Switching command with BRM | $30_{hex}$ | DO (fCI) | 1 | 1/5 |

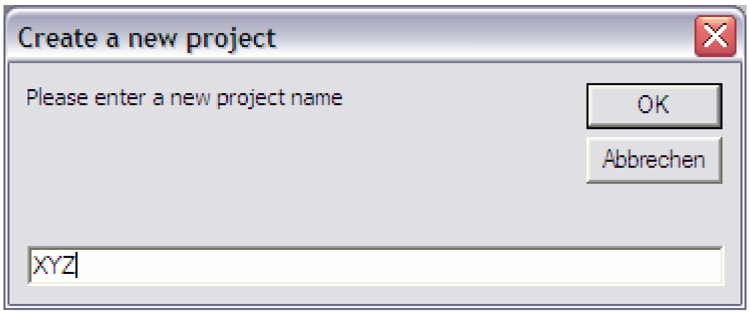
These function codes are used with 'bit splitting'.

More information about the card code, datapoint type and function code is given in Section 3.3.3.

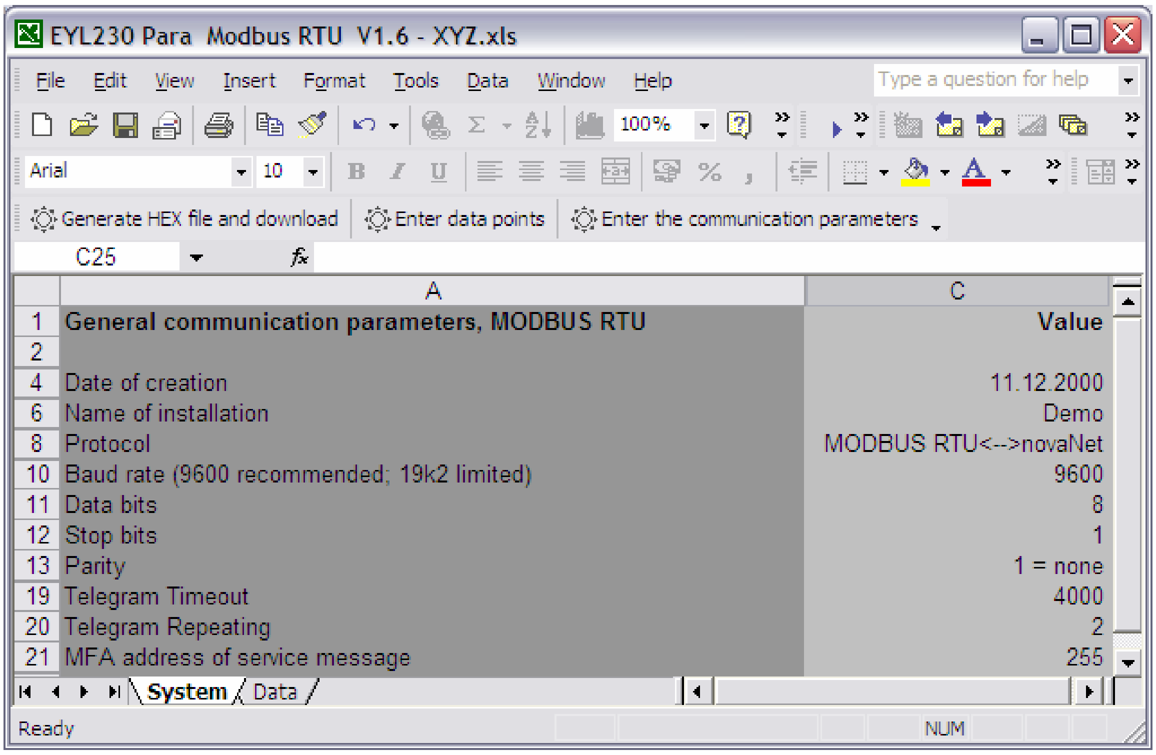
## 3.3 Engineering with ModbusPara230

### 3.3.1 Create a new project

- Start ModbusPara230 by opening file ModbusPara230.xls in the programme directory (Excel – File/Open/ModbusPara230.xls **or** Windows Explorer – double-click on ModbusPara230.xls)
- Click on 'Enable macros'
- Enter the new project name
  CAUTION: make sure that the selected name is not yet present in the programme directory.



After you enter the name, the newly created project is opened in Excel with these 2 tables:
- System
- Data


EYL230 Para Modbus RTU V1.6 - XYZ.xls
File Edit View Insert Format Tools Data Window Help
Generate HEX file and download
Enter data points
Enter the communication parameters
C25
System
Data
Ready
NUM

| | A | C |
|---|---|---|
| 1 | **General communication parameters, MODBUS RTU** | **Value** |
| 2 | | |
| 4 | Date of creation | 11.12.2000 |
| 6 | Name of installation | Demo |
| 8 | Protocol | MODBUS RTU<-->novaNet |
| 10 | Baud rate (9600 recommended; 19k2 limited) | 9600 |
| 11 | Data bits | 8 |
| 12 | Stop bits | 1 |
| 13 | Parity | 1 = none |
| 19 | Telegram Timeout | 4000 |
| 20 | Telegram Repeating | 2 |
| 21 | MFA address of service message | 255 |

The 'System' table contains the general communication parameters for the Modbus protocol and the hardware interface.
Each line, apart from the protocol line, can be modified.

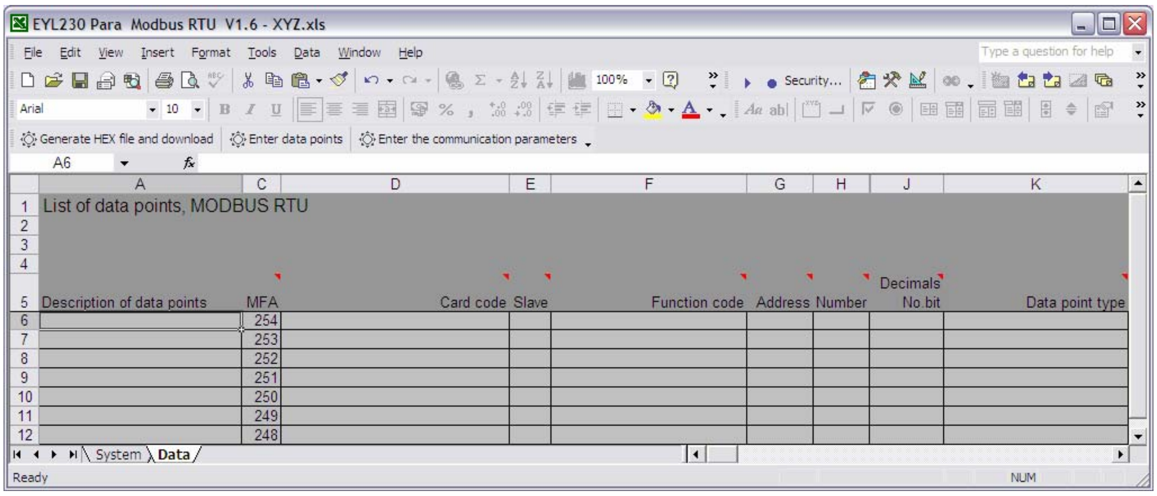
Notes:
Baud rate (Default value: 9600): Since protocol EPROM version b there is the possibility of a baud rate of 19'200. For technical reasons this transmission rate is not guaranteed, and functions just with some devices. The Modbus specification recommends a baud rate of 9600.

Telegram Timeout (Default value: 4000 ms): The protocol micro-programme sends cyclically a request telegram of all data points. If a data point does not answer within the specified telegram timeout the protocol micro-programme repeats the request for "Telegram Repeating"-times again.

Telegram Repeating (Default value: 2): Amount of repeats of request telegrams if response is wrong or missing.

The default values are in most cases optimal. There are Modbus devices and systems which are not optimized correctly with the default values. Further optimisation for system parameters (Character delay, telegram timeout after RX/TX error or after RX ok, "COM init" after x telegram errors) can be asked from Sauter (FAQ).

Below the icon bars, the following three buttons are available:
- Generate HEX file and download
- Enter datapoints
- Enter communication parameters



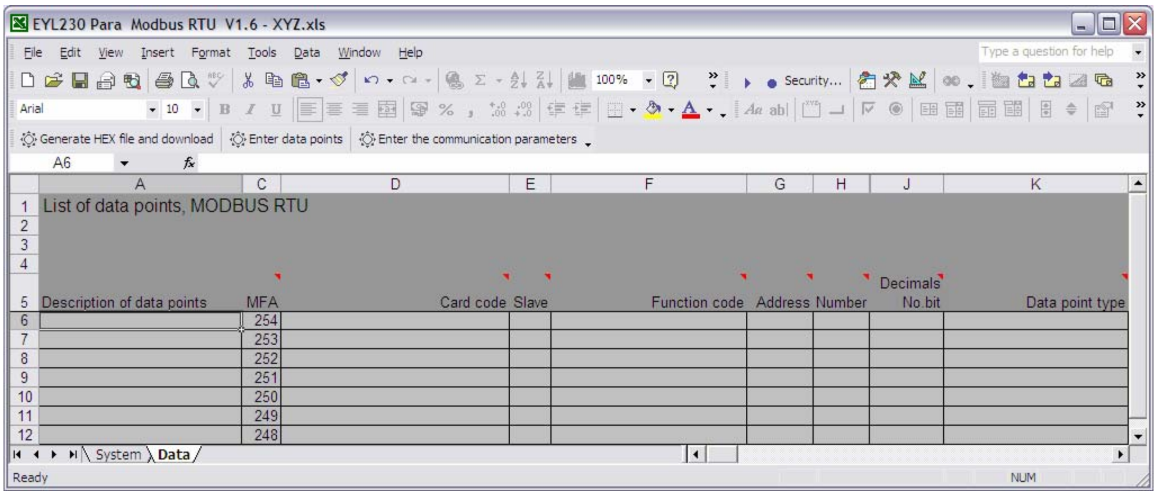
| | A | C | D | E | F | G | H | J | K |
|---|---|---|---|---|---|---|---|---|---|
| 1 | List of data points, MODBUS RTU | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | Description of data points | MFA | Card code | Slave | Function code | Address | Number | Decimals No.bit | Data point type |
| 6 | | 254 | | | | | | | |
| 7 | | 253 | | | | | | | |
| 8 | | 252 | | | | | | | |
| 9 | | 251 | | | | | | | |
| 10 | | 250 | | | | | | | |
| 11 | | 249 | | | | | | | |
| 12 | | 248 | | | | | | | |

After completing the datapoint list (Data table) and entering the communication parameters (System table), the HEX file can be generated and the file can be transmitted to the EYL230 automation station.

## 3.3.2 Open an existing project

- Open the project *projectname*.xls (for the definition of *projectname* see 3.3.1.: Create a new project) in the installation directory
(Excel – File/Open/*projectname*.XLS or, in Windows Explorer, double-click on *projectname*.xls)

## 3.3.3 Create or edit transfer list (datapoint list)


EYL230 Para Modbus RTU V1.6 - XYZ.xls
File Edit View Insert Format Tools Data Window Help
Generate HEX file and download
Enter data points
Enter the communication parameters
A6
System
Data
Ready
NUM

List of data points, MODBUS RTU

| | A | C | D | E | F | G | H | J | K |
|---|---|---|---|---|---|---|---|---|---|
| 5 | Description of data points | MFA | Card code | Slave | Function code | Address | Number | Decimals No.bit | Data point type |
| 6 | | 254 | | | | | | | |
| 7 | | 253 | | | | | | | |
| 8 | | 252 | | | | | | | |
| 9 | | 251 | | | | | | | |
| 10 | | 250 | | | | | | | |
| 11 | | 249 | | | | | | | |
| 12 | | 248 | | | | | | | |

**Datapoint description**
Description of the Modbus datapoint (may also, for example, coincide with the address text parameter for the corresponding MFA).

**MFA address**
MFA range: 254 - 64. The data must be entered continuously from MFA 254 onwards. There must be no gaps during entry, otherwise only the data up to the first empty field can be processed.
This means that there are a maximum of 191 MFAs available as transfer addresses.

## Card code

The card codes must match the Modbus datapoint types. This means, for example, that with analogue MODBUS datapoint types, analogue card codes and the corresponding analogue EY3600 modules have to be used. The table shows the possible combinations.

| Datapoint function | Hardware card code | CASE FBD modules | Datapoint type | Function code |
|---|---|---|---|---|
| Measured value | $70_{hex}$ | AI | 2 to 9 | 3, 4 |
| Metered value | $D0_{hex}$ | CI | 2 to 9 | 3, 4 |
| Setpoint | $80_{hex}$ | AO | 2 to 9 | 3/6 |
| Setpoint with ARM | $A0_{hex}$ | AO | 2 to 9 | 3/6 |
| Alarm / status // BRM | $10_{hex}$ | BI (fC8) // DI (fCI) | 1 | 1, 2, 3*), 4*) |
| Switching command | $20_{hex}$ | DO (fCI) | 1 | 1/5 |
| Switching command with BRM | $30_{hex}$ | DO (fCI) | 1 | 1/5 |

*)These function codes are used with 'bit splitting'.

If card codes without feedback ($80_{hex}$, $20_{hex}$) are used, the protocol micro-programme will not write the data that have been read to the MFA. To read switching commands or setpoint changes back into the MFA, this means that card codes with ARM ($A0_{hex}$) or with BRM ($30_{hex}$) have to be used.

## Slave

An address in the range between 1 and 247 must be assigned to each connected Modbus device.

## Function code

Describes the Modbus function that is used by the nova230. The function codes that are supported by the Modbus device must be identified from the instructions for the device. From the viewpoint of the nova230, there are various functions:

| | |
|---|---|
| function code 1/5 | = read/write digital values (Coils) |
| function code 2 | = read digital inputs (Discrete Inputs) |
| function code 3/6/16*) | = read/write analogue values (Holding Registers) |
| function code 4 | = read analogue inputs (Input Registers) |

Function codes 1 to 4 are executed cyclically by the protocol micro-programme and they read the values and inputs for the Modbus device. Function codes 2 and 4 are only read functions. Function codes 5 or 6 are used automatically by the protocol micro-programme and a Modbus telegram is sent if a command (DO) or a setpoint change (AO) is executed. Only one single value can be written.

CASE FBD modules DO ($20_{hex}$, $30_{hex}$) or AO ($80_{hex}$, $A0_{hex}$) can therefore ONLY be used together with functions 1/5 or 3/6 respectively.
On the other hand, BI ($10_{hex}$) or AI ($70_{hex}$) and CI ($D0_{hex}$) can also be used with function codes 1/3 or 3/6 respectively. In this case, function 3 or 6 is not used.

*)Function code 16 (Write Multiple Registers) is only used if a 32-bit analogue value is written. Other uses of the 'Write Multiples' function codes are not supported, and they are of little relevance for the AS.

No other Modbus function codes are supported.

**Address**
Defines the Modbus datapoint address of the datapoint. For analogue values, this is the word address (this address is used to address a 16-bit-wide memory cell). For digital values, this is the bit address (this address is used to address a 1-bit-wide memory cell).

Note: Modbus differentiates between addressing the Modbus PDU (Protocol Data Unit) with indices from 0 to 65535 and addressing the Modbus data model, which is numbered from 1 to n. The address range for nova230 is based on the indices of the Modbus PDU and is between 0 and 65535.
Addressing the Modbus data model is not precisely specified. Mapping of the Modbus data model on the nova230 Modbus addressing is entirely device-specific.
Manuals for Modbus devices often use addressing from 1 to n with a suffix that indicates the part of the data model from which the values come. For example: 0x : Bit values, 1x ; Discrete inputs; 3x : Input register; 4x : Register values. Accordingly, an address such as 4x0012 (often written as 40012) could be entered in the list with value 11 (omit the suffix and reduce the address by 1). See also [3] and the instructions for the Modbus device.

**Number**
Interrogation of several 16-bit registers (maximum 10), several 1-bit values (maximum 16) or several 32-bit values (maximum 5) in a telegram. The read registers must be assigned to an MFA in the subsequent lines. Care should be taken here to set the value in the 'Number' field to '0' in the following lines. This function can be used to optimise telegram interrogation, especially for larger systems.
With 16-bit and 32-bit registers, the registers have to be continuously. With 1-bit values it will read always with a 2 bytes telegram thus single bits can be read from a 16 bit block.

**Decimal point/bit number**
Decimal point: for analogue values (16-bit and 32-bit datapoint types), the value specifies the number of places after the decimal point. For example, a 16-bit signed with a decimal point = 1 will have a value range of -3276.7…+3276.7.
For digital values (datapoint type = bit and function code = 1/5 or 2) and for analogue floating-point values (datapoint type=8 or 9), the value must be set to 0.

Bit number: for digital values (datapoint type=bit, function code=3 or 4, BI ($10_{hex}$)) the value (0..15) specifies which bit is read from the 16-bit analogue value or the input - 'bit splitting'. Bit splitting is read-only, so it can only be used with card code $10_{hex}$.

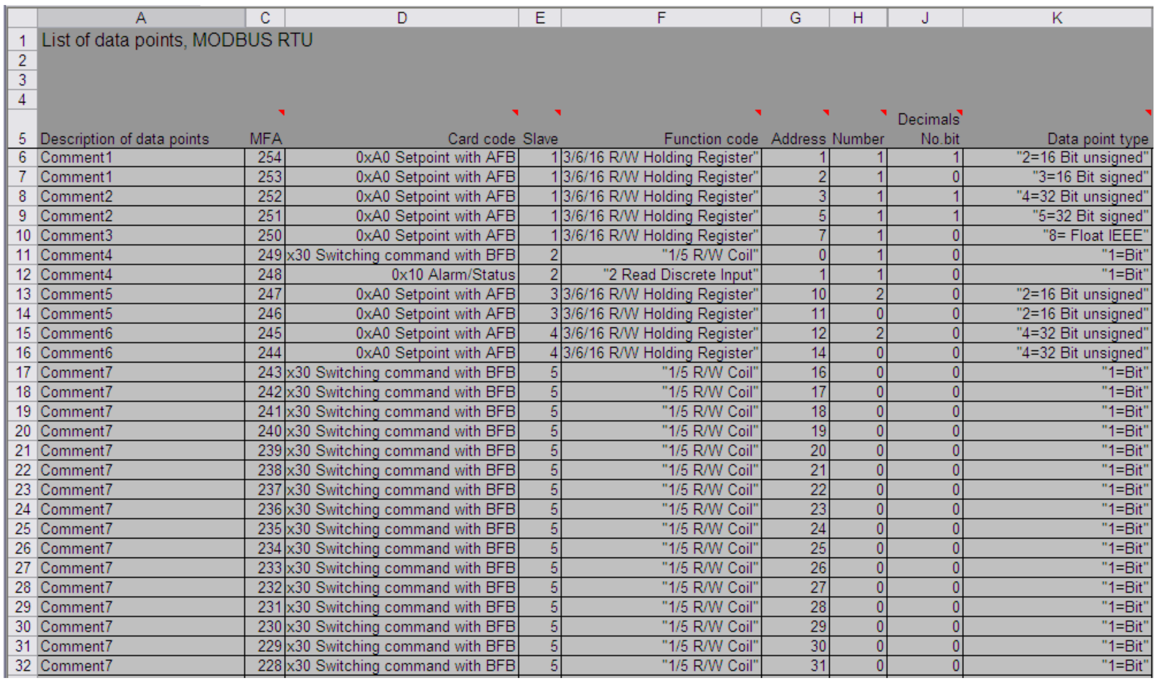
**Datapoint type**
The following datapoint types are supported:

| Datapoint type: | Value range: |
|---|---|
| 1 = Bit | 0, 1 |
| 2 = 16-bit unsigned | 0..65535 |
| 3 = 16-bit signed | -32767..32767 |
| 4 = 32-bit unsigned | $0..2^{32\ (28)}$ |
| 5 = 32-bit signed | $-2^{31\ (28)}..2^{31\ (28)}$ |
| 6 = 32-bit unsigned swapped | $0..2^{32\ (28)}$ |
| 7 = 32-bit signed swapped | $-2^{31\ (28)}..2^{31\ (28)}$ |
| 8 = IEEE floating point 32-bit | $3.4\text{x}10^{38}$ |
| 9 = IEEE floating point 32-bit swapped | $3.4\text{x}10^{38}$ |

32-bit datapoint types occupy two consecutive registers and are interpreted in the 'big-endian' (High/Low-Word) data format. The 'swapped' 32-bit datapoint types interpret two 16-bit registers in the 'little-endian' (Low/High-Word) format.

Note: only binary values can be interpreted. Other data types such as BCD-coded numbers, ASCII strings, etc. cannot be processed.

## 3.3.4 Example 1



| | A | C | D | E | F | G | H | J | K |
|---|---|---|---|---|---|---|---|---|---|
| 1 | List of data points, MODBUS RTU | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | Description of data points | MFA | Card code | Slave | Function code | Address | Number | Decimals No.bit | Data point type |
| 6 | Comment1 | 254 | 0xA0 Setpoint with AFB | 1 | 3/6/16 R/W Holding Register" | 1 | 1 | 1 | "2=16 Bit unsigned" |
| 7 | Comment1 | 253 | 0xA0 Setpoint with AFB | 1 | 3/6/16 R/W Holding Register" | 2 | 1 | 0 | "3=16 Bit signed" |
| 8 | Comment2 | 252 | 0xA0 Setpoint with AFB | 1 | 3/6/16 R/W Holding Register" | 3 | 1 | 1 | "4=32 Bit unsigned" |
| 9 | Comment2 | 251 | 0xA0 Setpoint with AFB | 1 | 3/6/16 R/W Holding Register" | 5 | 1 | 1 | "5=32 Bit signed" |
| 10 | Comment3 | 250 | 0xA0 Setpoint with AFB | 1 | 3/6/16 R/W Holding Register" | 7 | 1 | 0 | "8= Float IEEE" |
| 11 | Comment4 | 249 | x30 Switching command with BFB | 2 | "1/5 R/W Coil" | 0 | 1 | 0 | "1=Bit" |
| 12 | Comment4 | 248 | 0x10 Alarm/Status | 2 | "2 Read Discrete Input" | 1 | 1 | 0 | "1=Bit" |
| 13 | Comment5 | 247 | 0xA0 Setpoint with AFB | 3 | 3/6/16 R/W Holding Register" | 10 | 2 | 0 | "2=16 Bit unsigned" |
| 14 | Comment5 | 246 | 0xA0 Setpoint with AFB | 3 | 3/6/16 R/W Holding Register" | 11 | 0 | 0 | "2=16 Bit unsigned" |
| 15 | Comment6 | 245 | 0xA0 Setpoint with AFB | 4 | 3/6/16 R/W Holding Register" | 12 | 2 | 0 | "4=32 Bit unsigned" |
| 16 | Comment6 | 244 | 0xA0 Setpoint with AFB | 4 | 3/6/16 R/W Holding Register" | 14 | 0 | 0 | "4=32 Bit unsigned" |
| 17 | Comment7 | 243 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 16 | 0 | 0 | "1=Bit" |
| 18 | Comment7 | 242 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 17 | 0 | 0 | "1=Bit" |
| 19 | Comment7 | 241 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 18 | 0 | 0 | "1=Bit" |
| 20 | Comment7 | 240 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 19 | 0 | 0 | "1=Bit" |
| 21 | Comment7 | 239 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 20 | 0 | 0 | "1=Bit" |
| 22 | Comment7 | 238 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 21 | 0 | 0 | "1=Bit" |
| 23 | Comment7 | 237 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 22 | 0 | 0 | "1=Bit" |
| 24 | Comment7 | 236 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 23 | 0 | 0 | "1=Bit" |
| 25 | Comment7 | 235 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 24 | 0 | 0 | "1=Bit" |
| 26 | Comment7 | 234 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 25 | 0 | 0 | "1=Bit" |
| 27 | Comment7 | 233 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 26 | 0 | 0 | "1=Bit" |
| 28 | Comment7 | 232 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 27 | 0 | 0 | "1=Bit" |
| 29 | Comment7 | 231 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 28 | 0 | 0 | "1=Bit" |
| 30 | Comment7 | 230 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 29 | 0 | 0 | "1=Bit" |
| 31 | Comment7 | 229 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 30 | 0 | 0 | "1=Bit" |
| 32 | Comment7 | 228 | x30 Switching command with BFB | 5 | "1/5 R/W Coil" | 31 | 0 | 0 | "1=Bit" |

**Comments (see datapoint description column)**

1 Interrogation of a 16-bit signed or unsigned value.
This interrogation occupies 1 register in each case.

2 Interrogation of a 32-bit signed or unsigned value.
This interrogation occupies 2 registers in each case.

3 Interrogation of a floating point value (IEEE standard).
This interrogation occupies 2 registers in each case.

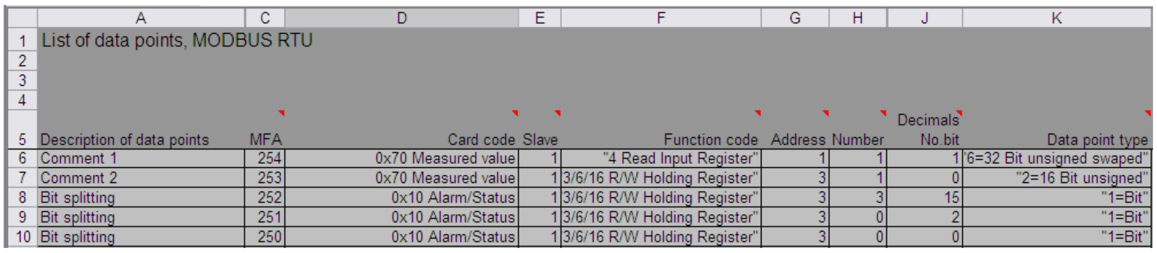
4 Interrogation of a bit from a specified bit address.

5 Interrogation of several 16-bit values in a telegram (a maximum of ten 16-bit values can be read in a telegram). The read registers have to be assigned to an MFA in the subsequent lines. It should be noted here that the number of datapoints is 0.

6 Interrogation of several 32-bit values in a telegram (a maximum of five 32-bit values can be read in a telegram). The read registers have to be assigned to an MFA in the subsequent lines. It should be noted here that the value in the 'Number' field is set to '0' in the subsequent lines.

7 Interrogation of several bit values from a specified bit address in a telegram (a maximum of 16 1-bit values can be read in a telegram). The read bit values have to be assigned to an MFA in the subsequent lines. It should be noted here that the number of datapoints is 0.

### 3.3.5 Example 2 – 'bit splitting'



| | A | C | D | E | F | G | H | J | K |
|---|---|---|---|---|---|---|---|---|---|
| 1 | List of data points, MODBUS RTU | | | | | | | | |
| 2 | | | | | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | Description of data points | MFA | Card code | Slave | Function code | Address | Number | Decimals No.bit | Data point type |
| 6 | Comment 1 | 254 | 0x70 Measured value | 1 | "4 Read Input Register" | 1 | 1 | 1 | '6=32 Bit unsigned swaped" |
| 7 | Comment 2 | 253 | 0x70 Measured value | 1 | 3/6/16 R/W Holding Register" | 3 | 1 | 0 | "2=16 Bit unsigned" |
| 8 | Bit splitting | 252 | 0x10 Alarm/Status | 1 | 3/6/16 R/W Holding Register" | 3 | 3 | 15 | "1=Bit" |
| 9 | Bit splitting | 251 | 0x10 Alarm/Status | 1 | 3/6/16 R/W Holding Register" | 3 | 0 | 2 | "1=Bit" |
| 10 | Bit splitting | 250 | 0x10 Alarm/Status | 1 | 3/6/16 R/W Holding Register" | 3 | 0 | 0 | "1=Bit" |

**Notes (see datapoint description column)**

1 Interrogation of a 'swapped' 32-bit unsigned value – 'little-endian'
This interrogation occupies 2 registers. The first register is the 16-bit Low-Word and the second one forms the 16-bit High-Word for the entire 32-bit value.

2 Interrogation of the 16-bit unsigned value for address 3.

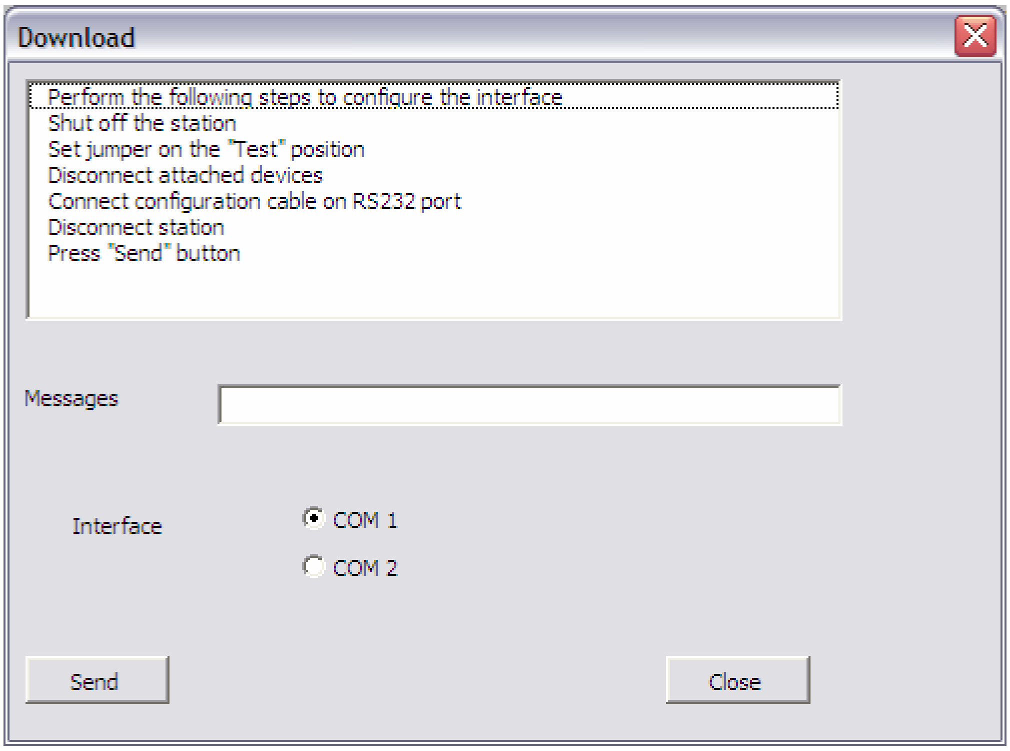
Bit splitting: interrogation of 3 bits (bit 0, bit 2 and bit 15) of address 3. In this case, the 3 bits are interrogated with one single telegram, since number=3 is set for bit 15 and number=0 is set for bit 2 and bit 0 .

### 3.3.6 Downloading the transfer list

- Switch the nova230 for Modbus (EYL 230 F040) off
- Move the jumper to the 'TEST' position
- Disconnect all Modbus components from the nova230
- Make the connection between the PC COM-Port and the serial connection (RS232) of the nova230 for Modbus (EYL 230 F040) on the 9-pin plug (RS-232) by using a zero modem cable or a novaNet291 cable
- Switch the nova230 for Modbus (EYL 230 F040) back on
- Use the 'Generate and download Hex File' menu item to generate the HEX file ...

HEX file has been successfully generated

- and to open the Download window.


Download

Perform the following steps to configure the interface
Shut off the station
Set jumper on the "Test" position
Disconnect attached devices
Connect configuration cable on RS232 port
Disconnect station
Press "Send" button

Messages

Interface [✓ COM 1] COM 2

- To start transmission of the generated transfer list into the nova230 for Modbus (EYL 230 F040), press the 'Send' button
- After about 2 minutes, the nova230 for Modbus is parameterised (green LED flashes)
- Wait until the 'Data transmission successful' message appears
- Switch the nova230 for Modbus (EYL 230 F040) off again
- Connect your Modbus components to the nova230 again
- Move the jumper to the 'RUN' position
- Switch the nova230 (EYL 230 F040) for Modbus back on
- After about 20 seconds, communication with the Modbus devices will start.

**Note:** during plant operation, the jumper must be in the '**RUN**' position!

### 3.3.7 LED status displays

**LED display, Modbus side**

| | | |
|---|---|---|
| Cycle LED (green) | lit | start phase |
| | flashing | cycle display while operation is running (jumper to 'Run') |
| Fault LED (red) | lit | fault |
| | in the start phase, this means: error in the micro-programme EPROM or in the USER-EPROM<br>during operation, this means: communication error (no response). | |

After you switch on, the Cycle LED is lit until the internal initialisation has been completed. Then it begins to flash.

### 3.3.8 Jumper position

**'Run' position**
Normal operation of the nova230.
The nova230 communicates with the Modbus components via the RS232 interface (DB9 plug).

**'Test' position**
With the jumper in the 'Test' position, telegrams and data logging can be enabled while operation is running. It should be noted here that the Fault LED is lit (continuously red). See Section 4, Monitoring mode.

If the jumper is in the 'Test' position when the nova230 is switched on, a menu will be outputted via the RS232 interface after initialisation. This menu can be used to enable the diagnosis outputs or to parameterise the station with the corresponding Modbus datapoints. (See Section 5: Troubleshooting).

# 4 Monitoring mode

For commissioning purposes, it is possible to create a monitoring circuit for a terminal programme on terminal 126 of the nova230.
The relevant connecting cable between the AS and the PC must be configured and the connection between the PC and the nova230 should be established as follows:

**Assignment:**

| **AS side (terminal)** | | **PC side (DB9 plug)** |
|---:|:---:|:---|
| | | 1 |
| 126 | ⇒ | 2 RXD |
| | | 3 TXD |
| | | 4 |
| ⊥ | ⇔ | 5 GND |
| | | 6 |
| | | 7 |
| | | 8 |
| | | 9 |

**Terminal settings:**

9600 Baud
8 data bits
no parity
1 stop bit
no handshake

To enable monitoring mode, switch the jumper over from the 'Run' position to the 'Test' position about 5 seconds after switching on.

A log from the terminal programme is shown below:

```
Request
 27.05 12:23.00
SlaveAddress=1 DP address=96 number=1 function code=1 bit
TX->010100600001FDD4
RX<-010101019048
value=       1   MFA=250  card code=0X20 value=  0.0
Request
 27.05 12:23.00
SlaveAddress=1 DP address=54 number=1 function code=4 IEEE Float 32-bit
TX->01040036000291C5
value=D0004196   MFA=249  card code=0X70 value=  0.0
Request
 27.05 12:23.00
SlaveAddress=1 DP address=68 number=1 function code=4 IEEE Float 32-bit
TX->01040044000231DEÿ
value=3800419E   MFA=248  card code=0X70 value=  0.0
Request
 27.05 12:23.00
SlaveAddress=1 DP address=22 number=1 function code=3 IEEE Float 32-bit
TX->01030016000225CF
RX<-010304000041A8CBDD
value=    41A8   MFA=254  card code=0X80
card code is not processed0X80
Request
 27.05 12:23.00
SlaveAddress=1 DP address=50 number=1 function code=4 IEEE Float 32-bit
TX->010400320002D004ÿ
value=    41A8   MFA=253  card code=0X70 value=  0.0
Request
 27.05 12:23.00
SlaveAddress=1 DP address=1 number=1 function code=2-bit
TX->010200010001E80A
RX<-010201016048
value=       1   MFA=252  card code=0X10 value=  80
```

# 5 Troubleshooting

If the jumper is in the 'Test' position after switching on, a menu is outputted via the RS232 interface after initialisation. This menu can be used to enable the diagnosis output, to list the datapoints, to read or delete the User-EPROM or to parameterise the nova230 with the relevant Modbus datapoints.
Troubleshooting uses the 9-pin RS232 C plug, with the help of a terminal programme.
Settings: 9600 Baud, 8 data bits, no parity, 1 stop bit.
To reach the parameterisation menu, switch the device off, set the jumper above the service socket to Test and switch the device back on.

After you press a button, you will see the parameterisation menu.

Terminal output:

```
        Parameterisation menu
1 = Parameterise configuration
2 = Parameterise datapoints
3 = List datapoints
4 = Diagnosis menu
5 = Write EPROM
6 = Read EPROM
7 = Delete EPROM
8 = Exit parameterisation menu
```

**Input control**
Use the Return key to confirm the entry.
You can use the backspace key to delete one character at a time. After you press Return to complete an entry, you can then only correct the value by going through the menu until you reach the same entry point.

**Description of the Parameterisation menu items**

**1 - Parameterise configuration**
You can enter a plant name and the date when the parameterisation was created here. You can also change the Baud rate and parity (default values: 9600 Baud, no parity) and the telegram waiting time (default value: 4000 ms).

**2 - Parameterise datapoints**
You enter the datapoints that are to be interrogated here. Start the entry with the MFA that you want to parameterise. 254 is outputted as the value. If datapoints have already been generated and more are to be generated and/or changed, you can enter the relevant MFA from which the parameterisation or change should take place.
After entering the MFA, enter the card code, slave address, function code, Modbus datapoint address, number of values to be read, number of places after the decimal point and the datapoint type.

For checking purposes, the dataset is outputted after the entry is completed. You can then finish the entry or add a new datapoint. To speed up input, the last indicated MFA is outputted as 1 lower. The old slave address is kept and the Modbus address is increased by 1. Press the Return key to confirm the entry.

The end of a list is reached by entering slave address 0. No datapoints parameterised after this point are taken into account, but they can be enabled at any time again by entering a slave address # 0 at the relevant point in the list. The datapoints must be entered continuously downwards from MFA254, without any gaps.

**3 - List datapoints**
All the enabled datapoints that have been parameterised can be listed again here.

**4 - Diagnosis menu**
The diagnosis meters can be interrogated here, or telegram logging and data logging can be enabled via the test interface.

The output from the diagnosis meters is split into function modules.

1. General diagnosis meters
   - protocol conversion
   - software version
   - reset meter
   - power-off meter
   - memory errors
   - number of memory initialisations
   - UP mapping errors
   - programme sequence errors
   - watchdog meter(s)

2. User-EPROM diagnosis
   - identification
   - protocol
   - plant name
   - date created

3. DMA diagnosis meters
   - number of memory initialisations for the internal memory map
   - number of memory initialisations for the external RAM
   - number of card code initialisations
   - number of deleted card codes
   - number of spontaneous responses
   - number of messages/signals
   - number of switching commands
   - number of setpoints
   - number of DMA read errors
   - number of DMA cycles

3. Driver diagnosis meters

- number of send telegrams
- " send errors
- " send repeats
- " receive telegrams
- " receive errors
- " receive repeats
- " initialisation errors
- " parity errors
- " break errors
- " POF errors

Error status bits

Bit 0 = 1 reception buffer free
Bit 1 = 1 telegram timeout
Bit 2 = 1 protocol sequence error
Bit 3 = 1 checksum error
Bit 4 = 1 character timeout
Bit 5 = 1 initialisation conflict
Bit 6 = 1 negative acknowledge
Bit 7 = 1 negative receive acknowledge
Bit 8 = 1 receive data while sending
Bit 9 = 1 parity/overrun or framing error
Bit10 =1 break
Bit11 =1 gateway error
Bit12 =1 transmitter active
Bit13 =1 waiting time over
Bit14 =1 acknowledge
Bit15 =1 gateway not ready

- " RxTx-reset meter
- " checksum error

4. Telegram diagnosis meters

- number of telegram timeouts
- " of interrogations
- " of responses
- " of telegrams with header errors
- " of acknowledgements
- " of data not entered
- " of unusable telegrams
- " of incorrect card codes
- " of commands

**5 - Write EPROM**
At present, the parameterisation is only stored in the RAM so it would be overwritten again if the device was restarted. This menu item must be enabled in order to write the data into the serial EEPROM.

**6 - Read EPROM**
Use this menu item to load the serial EEPROM into the RAM (read data out).

**7 - Delete EPROM**
All parameterised Modbus datapoints are deleted from the serial EEPROM.

# 6 Annexe

## 6.1 Connection assignment for connecting cables

### 6.1.1 nova230 <-> PC (Download)

Serial cable, DB9 socket to DB9 socket, crossed leads or rs-PARA-cable or novaNet291 cable.

**Assignment:**

| **AS side (DB9 socket)** | | **PC side COM1-Port (DB9 socket)** |
|---:|:---:|:---|
| 2 RXD | ⇐ | 3 TXD |
| 3 TXD | ⇒ | 2 RXD |
| 5 GND | ⇔ | 5 GND |

| **AS side Service interface (DIN plug, 5-pin, 180°)** | | **PC side COM1-Port (DB9 socket)** |
|---:|:---:|:---|
| 2 GND | ⇔ | 5 GND |
| 5 TXD | ⇒ | 2 RXD |
| 3 RXD | ⇐ | 3 TXD |

The communication parameters are set ex works.

### 6.1.2 nova230 <-> PC (monitoring)

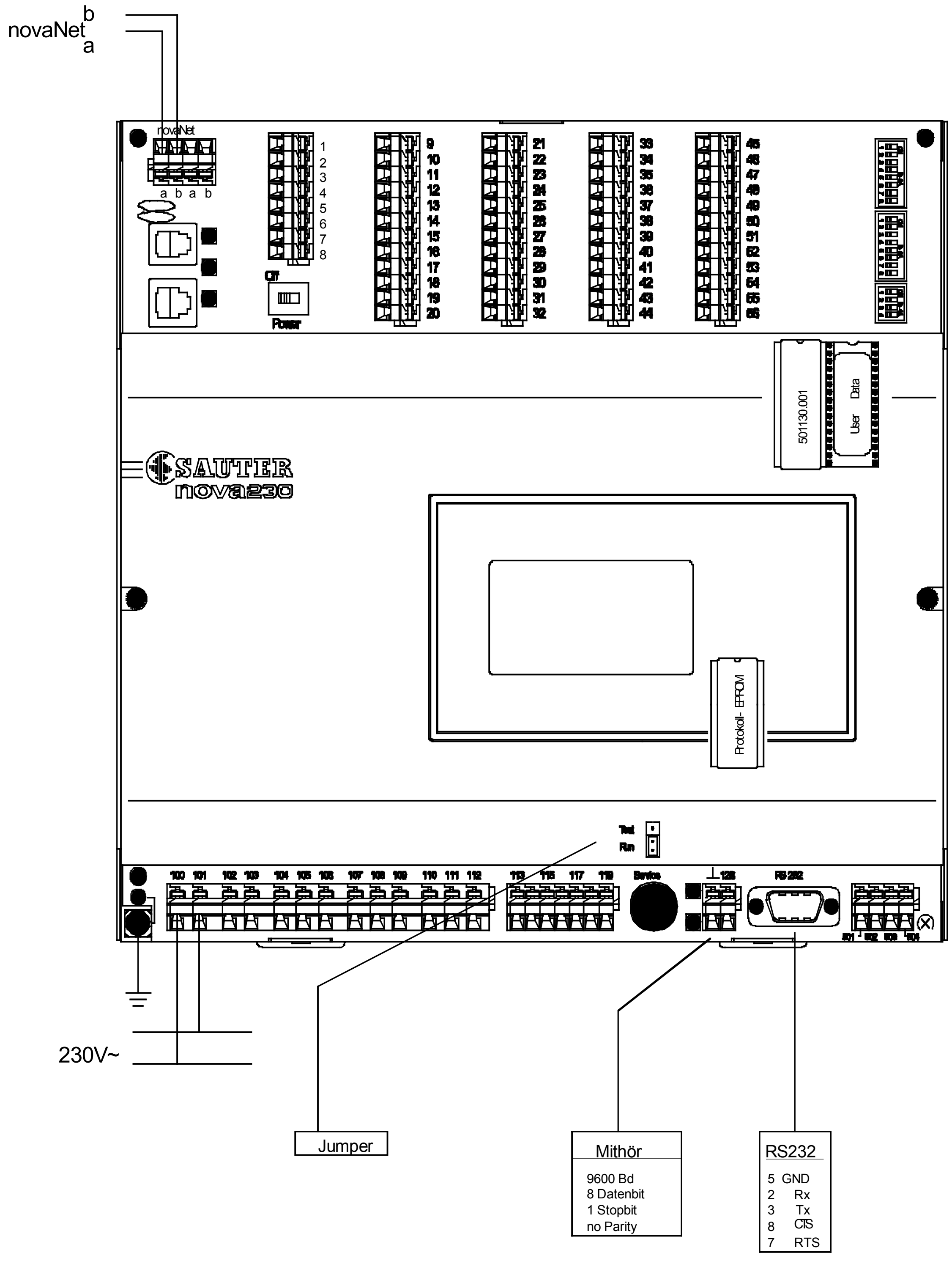
**Assignment:**

| **AS side (terminal)** | | **PC side (DB9 socket)** |
|---:|:---|:---|
| | | 1 |
| 126 | ⇒ | 2 RXD |
| | | 3 TXD |
| | | 4 |
| ⊥ | ⇔ | 5 GND |
| | | 6 |
| | | 7 |
| | | 8 |
| | | 9 |

**Terminal settings:**

9600 Baud
8 data bits
no parity
1 stop bit
no handshake

## 6.2 Connection drawing



## 6.3 Literature references and links

[1] MODICON Reference Guide PI-MBUS-300 Rev. B
[2] MODBUS over Serial Line, Specification & Implementation Guide V1.0
[3] MODBUS Application Protocol Specification V1.1

[4] www.modbus.org

The Modbus specifications can be downloaded officially from the Web site after personal registration.

## 6.4 Recommended auxiliary equipment

### 6.4.1 RS232/RS485 converter

Various RS232/RS485 converters are available on the market. Most converters function perfectly with nova230. Here are some converters that have been used with success (the list is not at all complete).

| Manufacturer | Product | Type | Source/contact |
|---|---|---|---|
| Phoenix Contact | Interface converter | PSM-ME-RS232/RS485-P | http://www.phoenixcontact.com |
| Wiesemann & Theis | Interface RS232 <> RS422/RS485, | #86201 | http://www.wut.de |
| ARP | Converter RS-232C-RS-485 | IC-485SI | http://www.arp.com |
| Wachendorff / Red Lion Controls | Interface converter ICM5 | ICM50000 | http://www.redlion.net |
| Omega | RS-232/485 Converter | LDM485-P | http://www.omega.com |

Note: we would be glad to add other products that have been successfully used to this list.

## 6.5 New functions

| Date - Version | protocol-EPROM: 501143.001 | Para-Programme | Changes |
|---|---|---|---|
| 2004 and earlier | Index a | 1.5 | |
| Aug 2005 | Index b | 1.6 | ▪ Bit splitting<br>▪ Word-Swapping<br>▪ Block reads of Bit values is corrected<br>▪ Translation of Modbus230Para to Italian<br>▪ Baud rate 19'200 with limitations<br>▪ Telegram Repeating<br>▪ Extensive Comments<br>▪ Manual Update (S8) |